University of Cape Town

Department of Computer Science

CSC3005h Class Test - Rewrite

2006

Marks : 35

Time : 45 minutes

Instructions:

- Answer all questions.
- Show all calculations where applicable.

Question 1: Symbol Tables and Activation Records [10]

a) What is a symbol table? [2]

a list of names and associated attributes

b) What is the difference between static and dynamic scope? [2]

static scope is defined by the lexical nesting of the program while dynamic scope is defined by the call sequence

c) Give an example of a language with only a single scope. [1]

assembler

d) For what types of programs do we NOT need to store activation records on a stack? [1]

no recursion

e) Assuming stack-based activation records, draw the full activation record stack corresponding to the function **not_main** at the position marked "%%%", as called by the function **main** in the following program: [4]

```
function main
start
   call not_main (1, 2)
stop
function not_main (x, y)
start %%%
   output (x, y)
stop
```

```
not_main: -----------------------+
    |      parm x=1              |
    |       parm y=2             |
    |       static_link ------------------+
    |       dynamic_link --------------+  |
    |       return address (main) |    |  |
main : --------------------------+ <--+  |
    |                            |       |
    +----------------------------+ <-----+
Minus one mark for each missing field.
```

Question 2: Intermediate Code [15]

a) Discuss 2 advantages of using intermediate representations. [2]

separation of front/back ends, easier to apply optimisations to

b) Using the attached IR language, convert the following C-like expression to an unoptimised IR tree. Assume **b** and **c** are stack variables at offsets *k_b* and *k_c* respectively from the frame pointer TEMP(FP). Assume **y** and **z** are constants. Provide the final tree and do not use the Nx/Cx/Ex expression types/objects. [4]

```
b = 2 * (y + z); c = (y + z)
```

*SEQ(MOVE(MEM(+(TEMP(FP),CONST(k_b))), *(+(CONST(y),CONST(z)),CONST(2))),*

MOVE(MEM(+(TEMP(FP),CONST(k_c))), +(CONST(y),CONST(z))))

Minus one mark for each major error.

c) Generate a new tree, applying common subexpression elimination as an optimisation. [4]

SEQ(

MOVE(TEMP(t), +(CONST(y),CONST(z))),

*MOVE(MEM(+(TEMP(FP),CONST(k_b))), *(TEMP(t),CONST(2))),*

MOVE(MEM(+(TEMP(FP),CONST(k_c))), TEMP(t)))

Minus one mark for each major error.

d) What is peephole optimisation? Give one example of peephole optimisation (besides the previous question). Give one example of a global/modular optimisation. [3]

peephole = when we optimise only small localised sections at a time e.g., constant folding [2]

e.g. of global = register allocation by graph colouring [1]

e) If a CJUMP is followed by its true label, what transformation can we make to the code so that it more accurately maps to actual machine code? [2]

Swap true and false labels and invert the conditional

Question 3: Code Generation [10]

a) Instruction selection is typically done by tiling. Name two algorithms used for tiling IR trees, and list their associated time complexities. Which of the two algorithms is generally slower and why? [5]

maximal munch O(N) [2]

dynamic programming O(N) [2]

dp, because the best solution is calculated for every node. [1]

b) When selecting instructions, it is better to use registers than memory access. Why? [1]

registers are local to the CPU

c) Briefly discuss 2 cases when data must be stored in memory instead of in registers? [2]

aggregate data structures, too many variables, etc.

d) Nested subprogram calls often lead to spilling of registers used for passing parameters. Briefly discuss 2 scenarios where such spills are not necessary. [2]

leaf procedures, different windows, done with variables, etc.